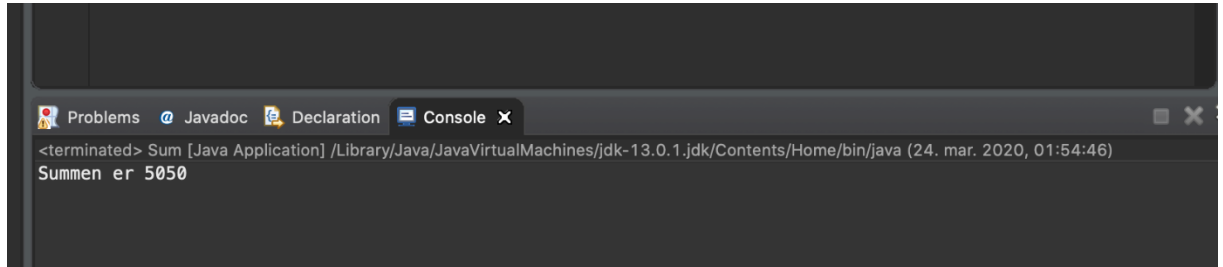


DAT102 Oblig 3

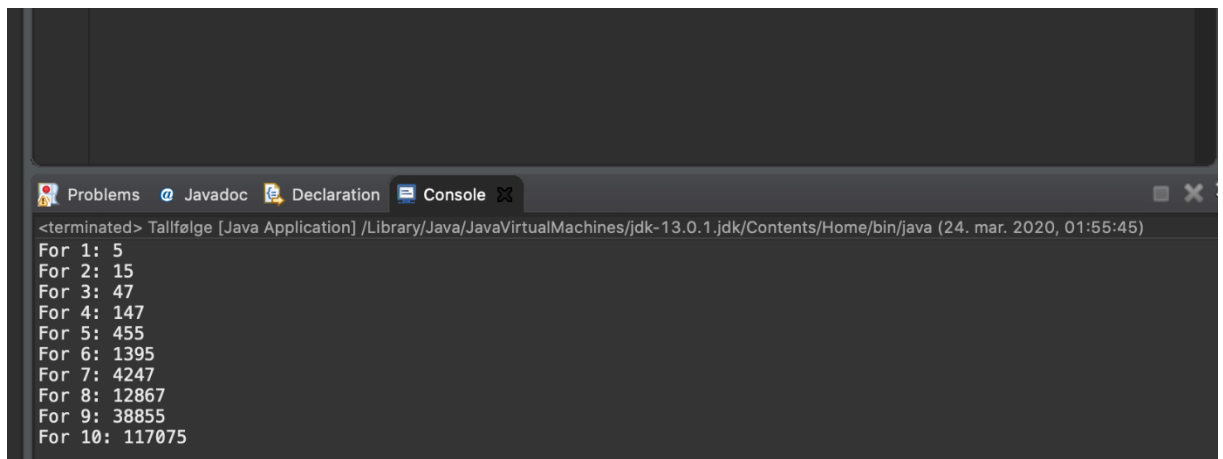
Oppgave 1

a)



```
<terminated> Sum [Java Application] /Library/Java/JavaVirtualMachines/jdk-13.0.1.jdk/Contents/Home/bin/java (24. mar. 2020, 01:54:46)
Summen er 5050
```

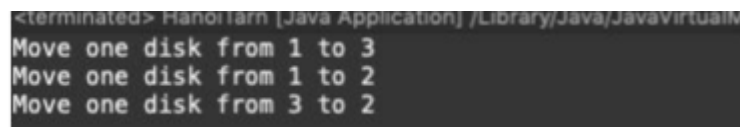
b)



```
<terminated> Tallfølge [Java Application] /Library/Java/JavaVirtualMachines/jdk-13.0.1.jdk/Contents/Home/bin/java (24. mar. 2020, 01:55:45)
For 1: 5
For 2: 15
For 3: 47
For 4: 147
For 5: 455
For 6: 1395
For 7: 4247
For 8: 12867
For 9: 38855
For 10: 117075
```

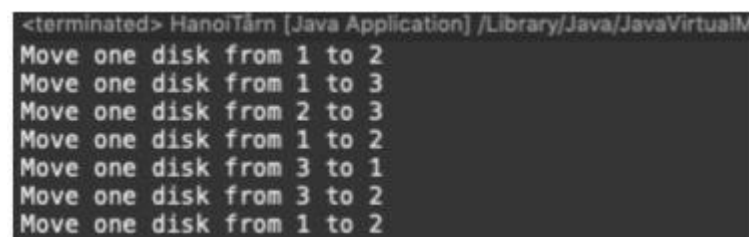
c)

N = 2



```
<terminated> HanoiTårn [Java Application] /Library/Java/JavaVirtualMachines/jdk-13.0.1.jdk/Contents/Home/bin/java (24. mar. 2020, 01:56:45)
Move one disk from 1 to 3
Move one disk from 1 to 2
Move one disk from 3 to 2
```

N = 3



```
<terminated> HanoiTårn [Java Application] /Library/Java/JavaVirtualMachines/jdk-13.0.1.jdk/Contents/Home/bin/java (24. mar. 2020, 01:57:45)
Move one disk from 1 to 2
Move one disk from 1 to 3
Move one disk from 2 to 3
Move one disk from 1 to 2
Move one disk from 3 to 1
Move one disk from 3 to 2
Move one disk from 1 to 2
```

N = 4

```
<terminated> HanoiTårn [Java Application] /Library/Java/JavaVirtualMachines/jdk-13.0.1.jdk/Contents/Home/bin/java (24. mar. 2020, 02:07:18)
Move one disk from 1 to 3
Move one disk from 1 to 2
Move one disk from 3 to 2
Move one disk from 1 to 3
Move one disk from 2 to 1
Move one disk from 2 to 3
Move one disk from 1 to 3
Move one disk from 1 to 2
Move one disk from 3 to 2
Move one disk from 3 to 1
Move one disk from 2 to 1
Move one disk from 3 to 2
Move one disk from 1 to 3
Move one disk from 1 to 2
Move one disk from 3 to 2
```

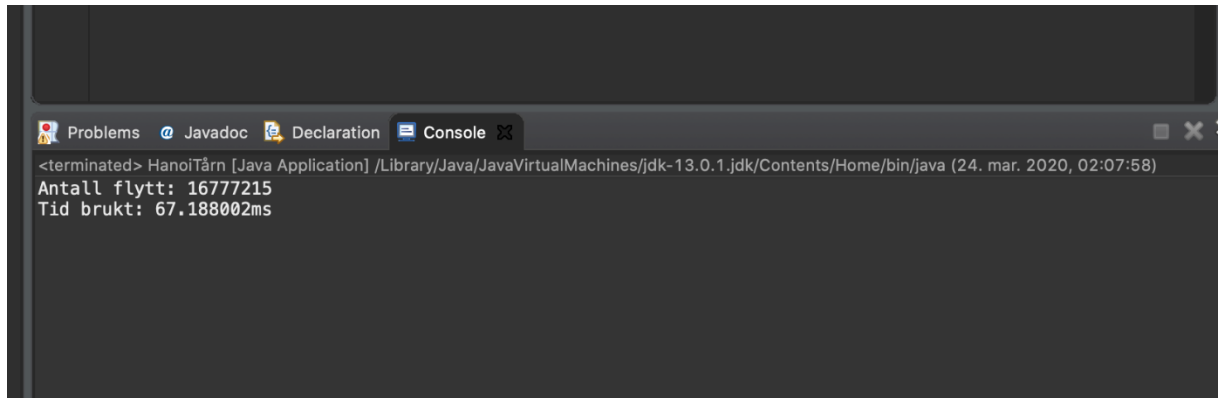
N = 20

```
<terminated> HanoiTårn [Java Application] /Library/Java/JavaVirtualMachines/jdk-13.0.1.jdk/Contents/Home/bin/java (24. mar. 2020, 02:07:18)
Antall flytt: 1048575
Tid brukt: 5.940968ms
```

N= 24

```
<terminated> HanoiTårn [Java Application] /Library/Java/JavaVirtualMachines/jdk-13.0.1.jdk/Contents/Home/bin/java (24. mar. 2020, 02:08:28)
Antall flytt: 268435455
Tid brukt: 609.467594ms
```

N = 28



The screenshot shows an IDE window with a tab labeled 'Console'. The console output is as follows:

```
<terminated> HanoiTårn [Java Application] /Library/Java/JavaVirtualMachines/jdk-13.0.1.jdk/Contents/Home/bin/java (24. mar. 2020, 02:07:58)  
Antall flytt: 16777215  
Tid brukt: 67.188002ms
```

iii)

Flytt for Tid_32 i programmet = 4294967295

Flytt for Tid_16 i programmet = 65535

Flytt for Tid_32 / tid_16 i programmet = 65537

$$2^{32} - 1 = 4\,294\,967\,295$$

$$2^{16} - 1 = 65\,535$$

$$4\,294\,967\,295 / 65\,535 = 65\,537$$

Oppgave 2

c)

Ved å oppdatere midten til $O(1)$ må vi oppdatere midten i leggTil-metoden. På denne måten oppdaterer man ut fra fire muligheter:

1. Listen er oddetall og man legger til venstre for midten. Midten blir oppdatert til å bli `midten.getForrige()`.
2. Listen er oddetall og man legger til høyre for midten. Midten forblir slik den er.
3. Listen er partall og man legger til venstre for midten. Midten forblir slik den er.
4. Listen er partall og man legger til høyre for midten. Midten blir oppdatert til å bli `midten.getNexte()`.

d)

i) Gjennomsnitt av antall noder man må gjennomgå er $n/2$, i verste tilfellet må man gå gjennom n antall noder.

ii) Her sammenligner man først med midtpekeren og finner ut hvilken halvdel man skal gå gjennom videre. Gjennomsnitt blir $n/4$, og verste tilfellet er $n/2$.

iii) Etter at man sammenligner med midtpekeren sammenligner man her med førstepeker, midtpeker og sluttpeker for å finne ut hvilken peker elementet man skal finne ligger nærmest. Det gir fire muligheter: søking fra førstepeker til midtpeker, fra midtpeker til førstepeker, fra midtpeker til sluttpeker, og fra sluttpeker til midtpeker. Gjennomsnitt her blir $n/8$, og verste tilfellet er $n/4$.

Oppgave 3

a)

Algoritme for binærsøking: $\text{int indeks} = -1$ $\text{int midtpunkt} = (\text{min} + \text{maks}) / 2$;

```
if (data[midtpunkt].compareTo(element) == 0) {  
    indeks = midtpunkt;  
} else if (data[midtpunkt].compareTo(element) < 0) {  
    if (midtpunkt + 1 <= maks) { indeks = binaersok(data, midtpunkt + 1, maks, element);  
    }  
} else if (min <= midtpunkt - 1) {  
    indeks = binaersok(data, min, midtpunkt - 1, element);  
}  
return indeks;
```

Dette er en algoritme for binærsøking som finner posisjonen for elementet man søker etter hvis det finnes, om det ikke finnes så -1.

b)

Liste: [2, 4, 5, 7, 8, 10, 12, 15, 18, 21, 23, 27, 29, 30, 31]

Skal finne 8. Finner midtpunkt $(14 + 0) / 2 = 7$

8 er ikke lik 15, går videre, finner ut at 8 er mindre enn 15

Christian Norill og Thomas Vu

Ny liste: [2, 4, 5, 7, 8, 10, 12], min = 0, maks = 6

Finner midtpunkt $(6 + 0) / 2 = 3$ 8 er ikke lik 7, går videre, finner ut at 8 er større enn 7

Ny liste: [8, 10, 12], min = 4 maks = 6

Finner midtpunkt $(6 + 4) / 2 = 5$ 8 er ikke lik 10, går videre, finner ut at 8 er mindre enn 10

Ny liste: [8], min 4, maks 4

Finner midtpunkt $(4 + 4) / 2 = 4$

8 er lik 8

indeks = midtpunkt = 4

returnerer 4

c)

Liste: [2, 4, 5, 7, 8, 10, 12, 15, 18, 21, 23, 27, 29, 30, 31]

Skal finne 16. Finner midtpunkt $(14 + 0) / 2 = 7$

16 er ikke lik 15, går videre, finner ut at 16 er større enn 15

Ny liste: [18, 21, 23, 27, 29, 30, 31] min = 8, maks = 14

Finner midtpunkt $(14 + 8) / 2 = 11$

16 er ikke lik 27, går videre, finner ut at 16 er mindre enn 27

Ny liste: [18, 21, 23] min = 8, maks = 10

Finner midtpunkt $(8 + 10) / 2 = 9$

16 er ikke lik 21, går videre, finner ut at 16 er mindre enn 21

Ny liste: [18], min = 8, maks = 8

Finner midtpunkt $(8 + 8) / 2 = 8$

16 er ikke lik 18, går videre finner ut at både midtpunkt + 1 ikke er mindre eller lik maks og min ikke er mindre eller lik midtpunkt - 1.

Returnerer -1

Overfor har vi fire sammenligninger.

$\log_2(16) = 4$

Dette stemmer for både antall rekursive kall og $\log_2(16)$.

Oppgave 4

Sortering ved innsetting

N	Antall målinger	Målt tid(gj.snitt)	Teoretisk tid $c \cdot n^2$
32000	10	1388.31ms	$1388.31 = c \cdot (32000^2)$ $c = 0.00000135616$
64000	10	5879.92ms	5554.83ms
12800	10	45644.77ms	22219,32544ms

Sortering ved utvalg

N	Antall målinger	Målt tid(gj.snitt)	Teoretisk tid $c \cdot n^2$
32000	10	712.92ms	$712.92 = c \cdot (32000^2)$ $c = 6,96210937e-7$
64000	10	3027.64ms	2851,68ms
12800	10	17256.12ms	11406,72ms

Boblesortering

N	Antall målinger	Målt tid(gj.snitt)	Teoretisk tid $c \cdot n^2$
32000	10	4257.96ms	$4257.96 = c \cdot 32000^2$ $c = 0.00000415816$
64000	10	17396.71ms	17031.82ms
12800	10	110264.23ms	68127.29ms

Kvikksortering

N	Antall målinger	Målt tid(gj.snitt)	Teoretisk tid $c \cdot n^2$
32000	10	17.57ms	$17.57 = c \cdot (32000 \cdot \log 32000)$ $c = 0.00012187441$
64000	10	26.99ms	37.49ms
12800	10	44.11ms	79.67ms

Flettesortering

N	Antall målinger	Målt tid(gj.snitt)	Teoretisk tid $c \cdot n^2$
32000	10	458.31ms	$458,31 = c \cdot (32000 \log 32000)$ $c = 0,00317907007$
64000	10	1576.02ms	977,87ms
12800	10	5924.92ms	2078,23ms

Radixsortering

N	Antall målinger	Målt tid(gj.snitt)	Teoretisk tid $c \cdot n^2$
32000	10	693.66ms	$693.66 = c \cdot 32000$ $c = 0.021676875$
64000	10	2689.42ms	1387.32ms
12800	10	12821.69ms	2774.64ms

c)

123	398	210	019	528	513	129	294
-----	-----	-----	-----	-----	-----	-----	-----

Fase 1:

0	1	2	3	4	5	6	7	8	9
210			123 513	294				398 528	019 129

Fase 2:

210	123	513	294	398	528	019	129
-----	-----	-----	-----	-----	-----	-----	-----

0	1	2	3	4	5	6	7	8	9
	210 513 019	123 528 129						398	294

Fase 3:

210	513	019	123	528	129	398	294
-----	-----	-----	-----	-----	-----	-----	-----

0	1	2	3	4	5	6	7	8	9
019	123 129	210 294	398		513 528				

Data sortert:

019	123	129	210	294	398	513	528
-----	-----	-----	-----	-----	-----	-----	-----

- d) Nye kvikksorteringen er raskere når du har lavere n-verdier. Ved høye n-verdier er den gamle kvikksorteringen raskest.